intel®

ARTICLE
REPRINT

AR-497

January, 1991

Designing With Flash Memory

By
MARKUS A. LEVY

Order Number: 295061-001

FEATURE ARTICLE

Markus A. Levy

Designing with Flash Memory

Is There a New Alternative to EEPROM and SRAM?

Flash memory (in general) is capturing market share from other memory technologies. It is replacing EPROMs that were traditionally used for code storage because, along with equivalent nonvolatility, it also allows in-system updates. Battery-backed SRAMs that once were used for data acquisition, parameter storage, and even solid-state disks are now targets for the inherently nonvolatile and lower-cost flash memory devices. Many notebook computer OEMs conclude that low power, light weight, and reliability are most easily obtained with a completely solid-state machine. Flash memory has achieved a density ramp from 256K bits to 2 megabits in two years. Combined with a special flash file system from Microsoft, flash memory can even replace the mechanical disk drive.

With the design described in this article, you have a platform demonstrating flash memory's functionality and flexibility. Applications range from data acquisition through an I/O port, to a DOS-compatible, solid-state disk. But first, a few essentials.

EPROM AND BEYOND

Derived from an EPROM process base, Intel's ETOX-II flash memory technology has similar nonvolatility, reliability, and array densities. In fact, the flash memory cell is identical to the EPROM structure, except for the thinner gate (tunnel) oxide. This is where the similarities end. The thinner gate oxide enables flash memory to be erased and reprogrammed in-circuit, typically 100,000 times. The name "flash" is derived from its one-second chip-level erase and microsecond-level byte-write times versus the slower, millisecond-level byte-write times for conventional EEPROMs.

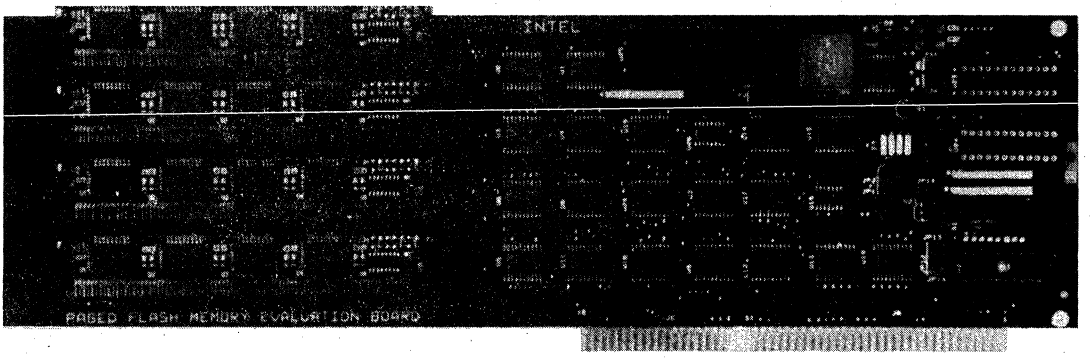
Flash memory devices have a command register architecture that provides a microprocessor-compatible write interface. Erase, program, verifications, and other operations are initiated by issuing the proper command to the flash memory device. Twelve volts must be applied on V_{PP} for the command register to respond to writes and execute the operation. The 12V requirement doubles as an added security feature for data integrity. If you are familiar with other memory subsystems, designing with flash memory is as simple as any other technology.

In addition to discrete components, Intel offers flash memory in SIMM and IC memory card formats. This design will use these modules, so I've included some pertinent information. The 512K-byte x 16 Intel Flash SIMM (SM28F001AX) is based on an 80-pin JEDEC standard that accommodates density upgrades and presence detect (a hard-wired ID that indicates SIMM density and speed). The eight 1-megabit flash memory devices on this module are paired up as high and low bytes. They are selected using the SIMM's write enable high and low (*WEH and *WEL) signals.

Intel's IC memory card adheres to the Personal Computer Memory Card International Association (PCMCIA) standard. This standard specifies physical, electrical, information structure, and data format characteristics of the card. Most impressive is the size, measuring 85.6 mm x 54.0 mm x 3.3 mm. Its 68-pin interface includes 26 address lines used to directly address 64 megabytes. All buffering and chip-level decoding is contained within the card, greatly simplifying the board-level design. Intel's flash memory card is available with one and four megabytes. These cards will continue to grow in density, becoming more and more competitive as disk drive replacements.

MEMORY METHODS

Three fundamental addressing methods can be implemented when interfacing a flash memory array to a system bus: linear, I/O, and paged. Each method has its benefits and drawbacks. A linearly addressed memory array is mapped directly into the sys-



tem's memory space and allows the highest performance. However, the memory array would be insufficiently small in systems having limited memory space, as with the 8086. But this method is practical in an 80386 (or other 32-bit processor) family system with a large memory space available.

An I/O-mapped memory array uses one address—an I/O port—to transfer data. This method requires the least amount of system memory space but also yields the lowest performance.

A page-mapped memory system is a hybrid of these two approaches. It allows a very large memory array with a minimal system interface. A page is a moveable window into the total memory array. It selectively opens different portions of the array by writing a page number to the decoding circuitry. This page ranges in size from 8K to 64K bytes. Analogous to a cache, a larger page size requires less frequent switching. Although switching pages represents a performance degradation, this can provide the optimal balance between performance and memory space availability within the system.

Our design is based on this page-mapped technique. A 64K-byte page size reduces the decoding circuitry. The PC/AT has been chosen as the execution platform, but with minor modifications to the control signals, any microprocessor environment can be used. Before beginning this design, it would be helpful to reacquaint yourself with the basics of the AT I/O channel bus.

The subsystems within this design (Figure 1) are the memory decode circuitry, I/O and its associated decode logic, and a 12V generator for V_{pp}. The Intel flash memory resides in four SIMMs. The board handles an upgrade path to 16M bytes, based on 4M-byte SIMMs.

ADDRESS DECODING

Flash memory addresses can be decoded in one of two ways: row-column and conventional decoding using separate chip enables. The row-column approach of Figure 2 is appropriate if you are motivated to reduce board traces. In row-column decoding, rows are Output Enables (*OE), Write Lows (*WRL), and Write Highs (*WRH); and columns are Chip Enables (*CE). Although the SM28F001AX uses only four chip enables, eight are provided since four-megabyte SIMMs could consist of sixteen 2-megabit flash memory devices.

Page selection, discussed in more detail later, is accomplished by writing the page number through an 8-bit I/O port to a latch. This will allow access to 256 64K-byte pages. Page signals, P0–P2, are directly connected to A15–A17 on each SIMM. They decode pages on the device level. The row-column signals are derived by decoding page signals P3–P7. They enable components on the SIMMs.

The row-column approach, however, suffers from simultaneous selec-

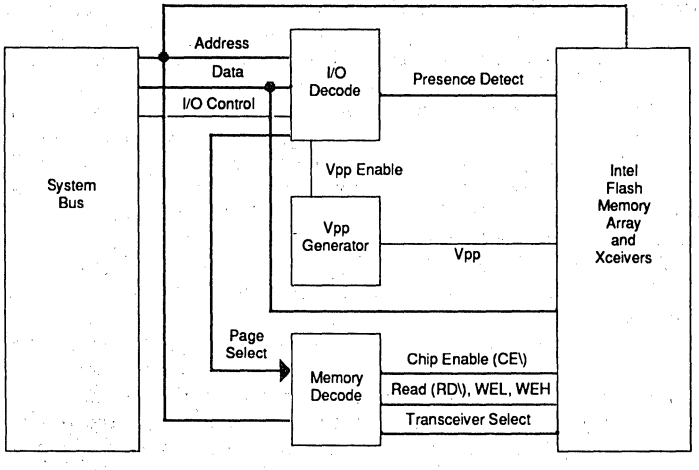


Figure 1—*The subsystems in a flash memory board design include memory decode, I/O, and a 12V generator.*

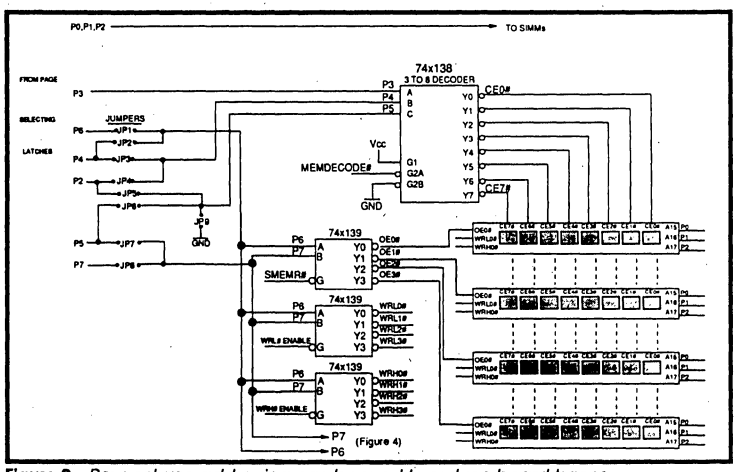


Figure 2—*Row-column addressing can be used to reduce board traces.*

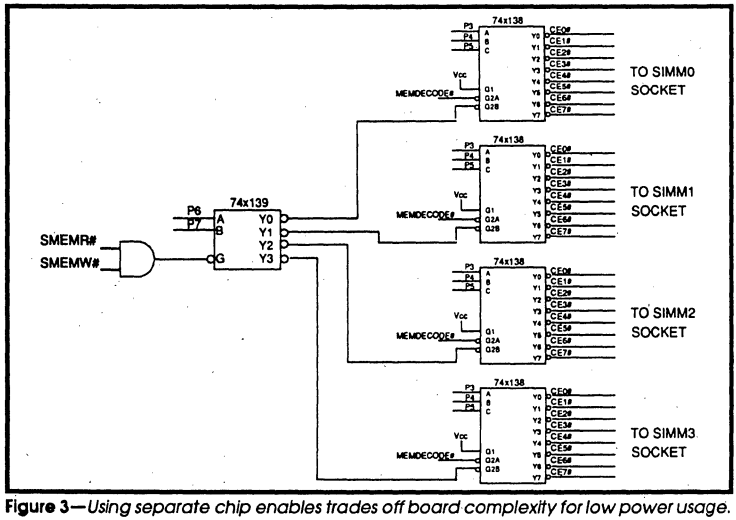


Figure 3—*Using separate chip enables trades off board complexity for low power usage.*

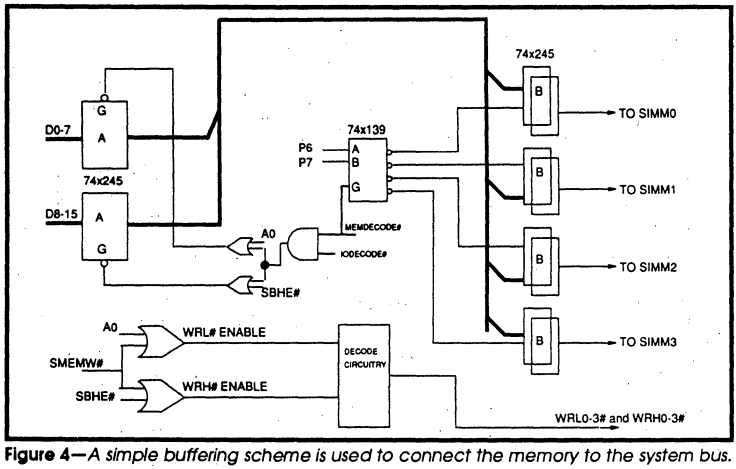


Figure 4—*A simple buffering scheme is used to connect the memory to the system bus.*

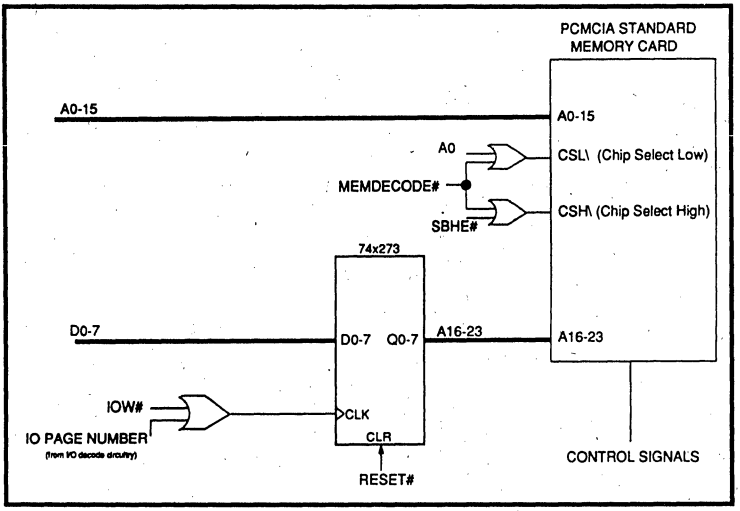


Figure 5—*Moving some buffering onto a separate memory card reduces the amount of necessary interface hardware.*

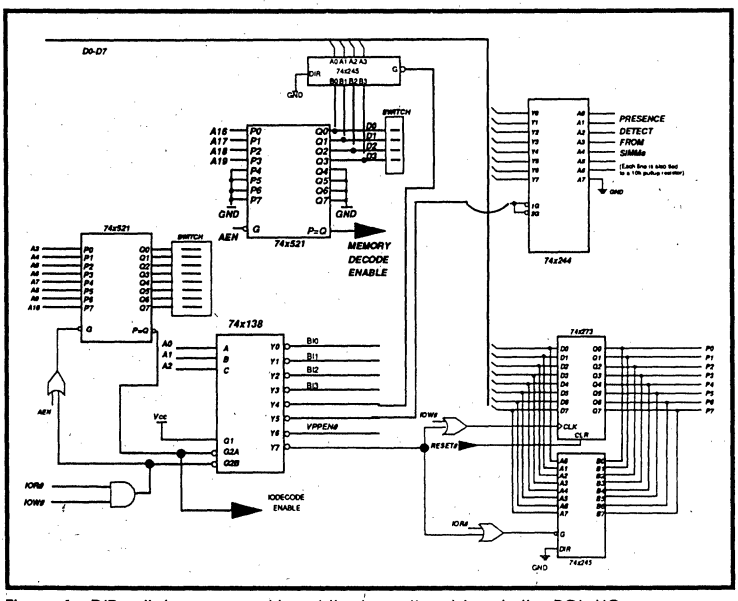


Figure 6—*DIP switches are used to set the board's address in the PC's I/O space.*

tion of eight parallel device chip enables. The conventional, separate chip enable decoding method has lower power consumption. Using this method (Figure 3), upper page signals, P6 and P7, decode which SIMM is selected.

Page selection becomes more complicated when accounting for density upgrades. To understand this there are two things to keep in mind: each SIMM socket handles a maximum of 64 pages, (i.e., 4-megabyte SIMMs), and A17 is used on a 2-megabit device but is a "No Connect" on the 1-megabit part. A "No Connect" pin implies that page selection will not be contiguous with SIMMs less than four megabytes in size. Accommodating noncontiguous pages increases software complexity. Regardless of the page decoding method, a jumper scheme rearranges the page signals and accommodates density upgrades.

According to Figure 2, the jumper settings are as follows:

1-MByte SIMM—J7, J2, J9, J4
2-MByte SIMM (16 x 28 F010)—J7, J1, J3, J5
2-MByte SIMM (8 x 28 F020)—J7, J1, J9, J3
4-MByte SIMM (16 x 28 F020)—J8, J1, J6, J3

Figure 4 shows the buffering required for system bus interfacing. The PC I/O channel bus is limited to two TTL loads on any one line. The "A" transceivers are connected directly to the I/O channel bus. Additionally, each SIMM has its own pair of "B" transceivers to reduce capacitive loading that results from tying more than eight flash memory devices together.

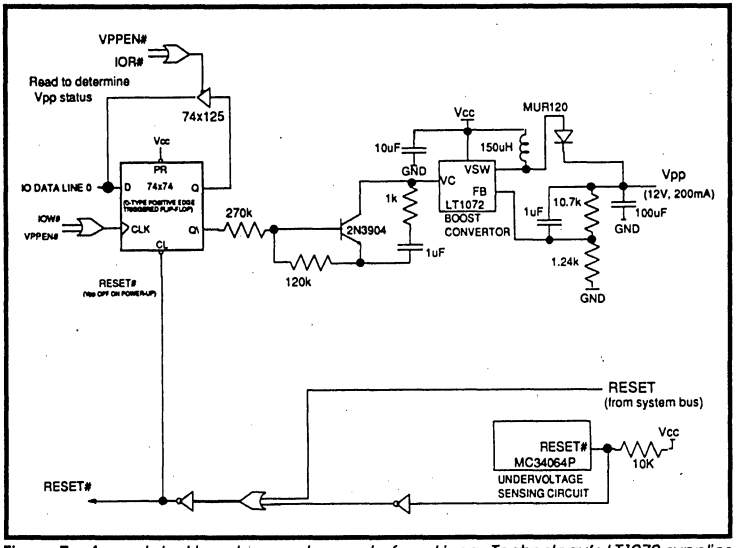


Figure 7—*A regulated boost converter made from Linear Technology's LT1072 supplies the 12 V necessary for programming.*

SIMPLIFIED DECODING HARDWARE

You are not limited to using SIMMs in your design. The same basic techniques will work for discrete components or other module types, such as memory cards. The IC memory card provides the simplest solution with its integrated decoding. Writing the page number through an I/O port to a latch accomplishes the same page selection as before, but now the "data" translates directly into memory card address inputs A16–A23. This eliminates the entire decoding structure shown in Figure 2 or 3. The IC card itself also contains the "B" transceiver buffering. This results in the hardware reduction shown in Figure 5. You will want to purchase a spring-loaded connector from Fujitsu, AMP,

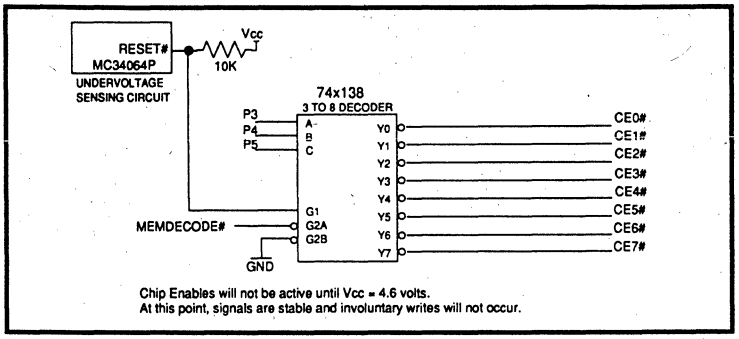


Figure 8—*An MC34064P undervoltage sensor is used to disable all writes to memory when the power starts to fail.*

or ITT Cannon to use with the memory card, laying out your board so that the memory card can be retrieved out the back of your PC. Pushing the button on the connector ejects the memory card for data security or transport.

I/O PORTS ON THE PAGE MEMORY BOARD

The page-selecting I/O port mentioned is one of eight ports in this page memory board design. The 74*x*521 comparator's inputs begin with A3 to simplify the decoding circuitry for the I/O port addresses. This places the base I/O port on an even 8-byte boundary. Use discretion when setting the switch to avoid conflict in the AT environment. I recommend using I/O addresses between 300H and 318H because this area is assigned for prototype cards.

Listed in the order in which they appear on the 74*x*138 decoder in Figure 6, the eight I/O ports in this design are: BI0–BI3 (accesses the board's identifiers); the window address within the system's memory space; the SIMM's presence detect; the V_{pp} enabling register; and the page number reading and writing ports.

I didn't diagram the board's identifier circuitry because the implementation is extremely simple. It consists of four identical units made up of a 74*x*244 and an 8-input DIP switch. The four enables, BI0–BI3, from the I/O decoder connect to the corresponding unit's enable. The I/O space is scanned for the identifiers to locate the board.

The user-selectable page window address can be placed on any 64K-byte boundary within the DOS 1-megabyte range, but only the adapter ROM area between C0000H and E0000H will alleviate compatibility problems. Using additional inputs on the "Memory Decode Enable" 74*x*521 and an 8-input DIP switch allows window placement above 1M byte. Simply connect address lines A20–A23 to input pins P4–P7, respectively. The Presence Detect (PD) pins from all four SIMMs are wire ORed together into the 74*x*244. This eliminates having to read each SIMM's PD pins separately, but SIMMs must be installed with equivalent density and speed.

The page number is written and read through the same I/O port. Minimal circuitry ensures that the system powers up at page zero. The page memory board's *RESET signal connected to the CLR input of the 74*x*273 latch accomplishes this task.

V_{pp} GENERATION

Why is V_{pp} generation necessary on this flash memory board when the AT I/O channel already has a 12-V supply? True IBM-compatible PCs specify the necessary ±5% 12-V supply tolerance. However, some systems have wider 12-V supply tolerances. Local 12V generation via the circuitry described below ensures fast, reliable flash memory operation.

Linear Technology's LT1072 switching regulator, a 5-V–to–12-V feedback regulated boost convertor, is the heart of the V_{pp} generator (Figure 7). A 100-µF capacitor at the output handles up to 200 mA, necessary for software that programs or erases

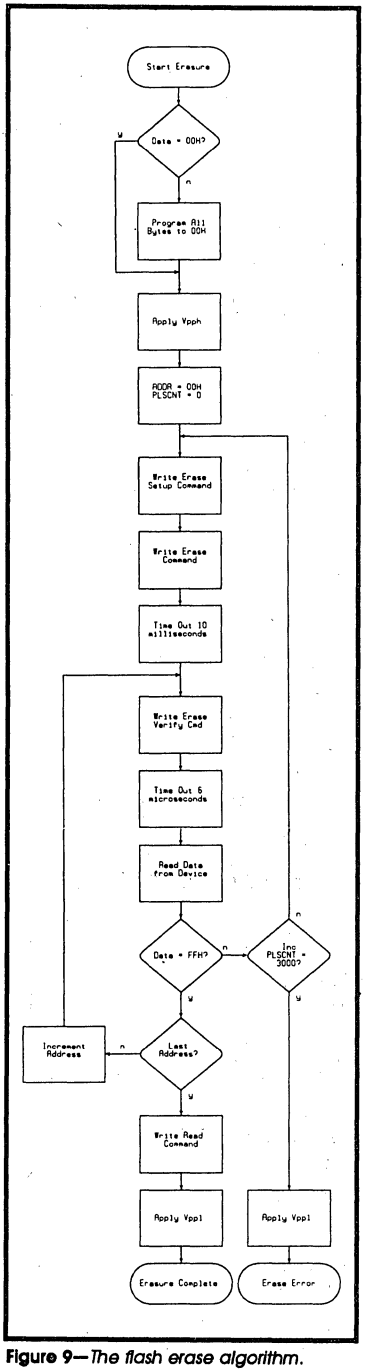


Figure 9—*The flash erase algorithm.*

eight flash devices simultaneously. Turning V_{PP} off when not in use conserves power, but this capacitance value requires approximately 100 ms to fully charge to 12 V. Reducing the output capacitance value and limiting the number of flash memory components accessed simultaneously decreases the ramp time. The diode, MUR120, prevents inductor current absorption from the charged output capacitor. During system power-up, spurious noise may generate writes which are actually the sequence of flash memory commands that initiate erasure or programming. Disabling V_{PP} until voltages stabilize provides power-up protection. The Motorola MC34064P is an undervoltage sensing circuit that begins functioning when V_{CC} is above 1 volt. Between 1 and 4.6 volts, the MC34064P's *RESET output or AT system RESET clears the 74x74. While the 74x74 remains cleared (or *Q = 1), the 2N3904 is on, the VC input of the LT1072 is 0 volts, and the VOLTAGE SWITCH (VSW) output is off. Writing a one to the V_{PP} enable latch forces *Q low, turning off the transistor. This puts the VC input at 5 V, and VSW output generates 12 V.

You do not need the circuitry just described if your system's 12-V supply meets the V_{PP} specifications. However, because software may accidentally (or coincidentally) generate a valid flash memory command to a flash memory address, install a switch to turn off V_{PP} when not in use. A low-resistance PFET (Motorola MTD4P05) performs this duty.

The possibility of spurious writes to the flash memory devices during power-up still exists. Again, the same undervoltage sensor (MC34064) solves this problem. The *RESET output becomes the 74x138 decoder's active-high enable. This controls the chip or write enables for the flash memory devices (Figure 8).

A FEW ADDITIONAL POINTERS

Ground *MEMCS16 so the PC/AT recognizes your board with a 16-bit bus width. The original design operated in both 8- and 16-bit systems. This flexibility is accomplished with additional decoding that multiplexes the high data bus onto the lower data bus. Again, the IC memory card automatically conforms to either bus width because the extra decoding is handled internally.

As with any circuit design, it is important to follow good design principles. For example: decouple power supplies with 0.1-µF capacitors between V_{CC} and V_{SS} of every device; and short board traces help minimize noise.

SOFTWARE

Hardware without software is like a computer without a processor. Therefore, understanding program and erase algorithms is the first step towards functional flash memory. Recall from our earlier discussion that operations on flash memory are software controlled using the internal command register architecture. I have included the complete algorithms (Figures 9 and 10). *[Editor's Note: Software for this article is available on the Circuit Cellar BBS and Software On Disk #18. See page 92 for downloading and ordering information.]* After working the Intel flash memory hotline, I would like to discuss the common mistakes.

The best piece of advice that I can give is *please* follow the algorithms. Many people unsuccessfully try their own custom versions. There are no cutting corners. Starting with the basics, let me elaborate a few points about the algorithms:

1) Flash is programmed to a binary zero by adding charge to the transistor's floating gate. Contrarily, charge removal erases the cell to a binary one. During the erase operation, the device is "flashed" as charge is simultaneously and equally pulled off the floating gate of every memory cell. The device must be preprogrammed to all zeros before erasure so an already erased cell is not further depleted of charge.

2) Prior to writing any command, switch on V_{PP} and allow ample ramp time for proper operation. Dropping V_{PP} below 7.5 volts from within any operation, places the device in the read mode. Similarly, abort an operation by issuing two consecutive reset commands (FFH) followed by the read command (00H).

3) Closely observe delay times to achieve the highest performance and

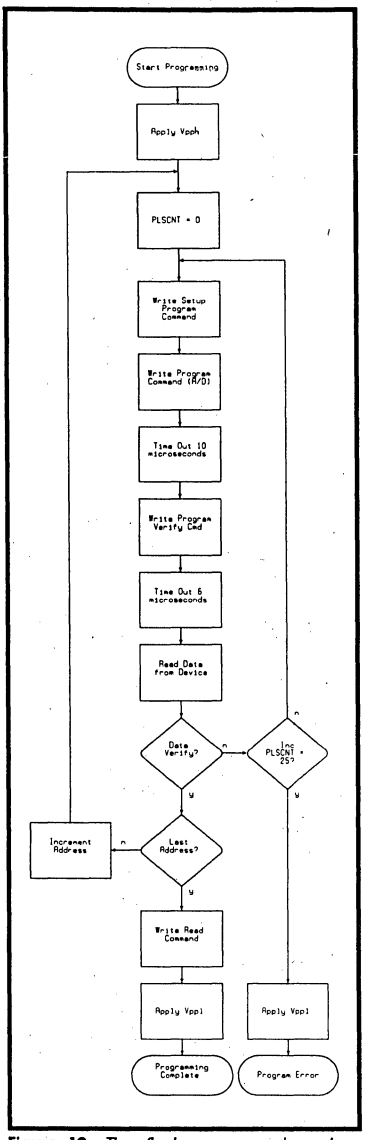


Figure 10—*The flash programming algorithm.*

reliability. Use the `STI` instruction in the software drivers to avoid system interrupts during these delays. Execute `CLI` once the corresponding verify command is issued.

4) The verify operation internally creates marginal conditions to ensure accurate and reliable results. The 6-µs slew time delay following the verify command allows the margin voltages to settle. In the verify mode, programmed data is guaranteed to be "permanent" when it matches the data being programmed.

USING YOUR PAGE MEMORY BOARD AS A DATA RECORDER

There is a nearly endless list of data recording applications, including digital imaging, digital photography, point-of-sale terminals, patient monitors, and flight recorders. The page memory board is appropriate for many data recording applications where an I/O port of the PC/AT accumulates data.

The programming algorithm demonstrates the byte programming capability of flash memory. In other words, once the device is erased, bytes reprogram randomly. In some recording applications, data is received in packets. A pointer determines where to begin programming the next free location within the flash memory.

Interleaving increases write performance by using the idle time during the 10-µs program delay. Addresses are offset such that each successive data byte gets written in different devices, looping back in time to issue the verify command. Reading the data back would be done in a similar fashion.

Word-wise, or parallel, programming of two devices provides an additional means of increasing write performance. Note that flash memory devices may program at different rates. Therefore, the original algorithm must be modified during the verify operation. If only one byte of the word has verified, the program command and data are sent again to the unverified byte. Mask the command sent to the device that verified to maintain word-wise programming. A mask is the substitution of a reset command for the program and verify commands. That way, the programmed bytes do not get further programmed on subsequent pulses.

The page memory board will perform these software techniques. However, first write the software that determines the location and capacity of the board. First, scan the I/O space for the board's identifier. The location of the first identifier byte is also the

base address for the eight I/O ports. Using the proper offset, read the I/O port that enables the base-memory address transceiver. For SIMMs, calculate the memory capacity by first reading the Presence Detect pins, followed by reading the individual flash memory device identifiers. Alternatively, read the Card Information Structure in the PCMCIA standard memory card for the capacity.

The preceding steps will confirm the basic functionality of your hardware. Practice programming the flash devices with data from a RAM-based array. For example:

```
; Software to read in ASCII test
; pattern to program into flash
DATA_ARRAY SEGMENT
STORE_IN_FLASH
    DB  'ASCII test pattern to
    DB  'be stored in flash'
DATA_ARRAY ENDS

CODE SEGMENT
    mov  ax,DATA_ARRAY
    mov  ds,ax
    mov  si,0
    mov  cx,size STORE_IN_FLASH

more_input:
    mov  al,[si]
    call FLASH_PROGRAM
    inc  si
    loop more_input
```

Now the fun begins. Once you've put your Flash Paged Memory Board together and tried it out, imagine converting it into a solid-state disk. It's been done using software drivers from Microsoft. In an upcoming article, we will discuss the structure of the Microsoft Flash File System and show you how to interface it to your board.

Finally, I would strongly suggest that before attempting this design, you obtain the appropriate flash memory literature from Intel (see the source box).

Markus Levy is an application engineer at Intel Corporation in Folsom, California, and holds a B.S.E.E from California State University. His specialties include software and hardware implementations of solid-state disks in portable computers. His favorite away-from-work pastimes include home remodeling, swimming, and parenting.

SOURCE

Intel Literature: (800) 548-4725

Data sheets

SM28F001AX	1 M-byte SIMM	#290244
1MC001FL	1 M-byte IC memory card	#290399
4MC001FL	4 M-byte IC memory card	#290388
28F020	2 M-bit device	#290245

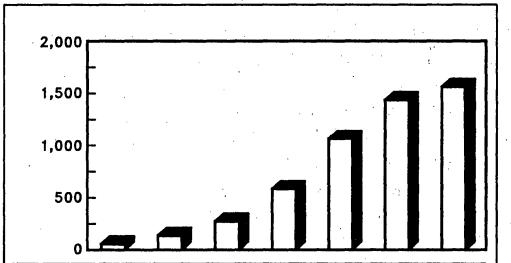
Application notes

AP325, Guide to Flash Memory Reprogramming	#292059
AP343, High-Density Applications Using Intel Flash Memory	#292079

BOOT BLOCK FLASH: THE NEXT GENERATION

INTRODUCTION

Flash memory offers a whole new dimension to nonvolatile memory applications because of its high density and cost-effectiveness. Embedded systems, such as PC BIOS, hard disk drive controllers and laser printers, were among the first applications to utilize the easy update capability of Flash, allowing system differentiation and upgradeability. In 1992, Intel introduced a series of high-density Flash Memory Cards based on its 8-Mbit FlashFile™ component, products that are enabling truly portable computing capability with diskless, solid-state mass storage. Intel also added two flash memory products designed specifically to serve more traditional embedded and portable applications, offering both high performance and low power consumption options in an architecture specifically designed for these applications storage requirements.



	1990	1991	1992	1993	1994	1995	1996
$MM	35	131	278	626	1,104	1,449	1,632

297137-1

Source Dataquest: January 1992

Figure 1

MARKET GROWTH DATA

In the four years since Intel first introduced its 256-Kbit flash devices, the worldwide demand has quadrupled annually, with 1992 market demand forecasted to exceed $270 million. Flash is expected to continue double-digit growth through the upcoming years while other memory technologies are relatively flat or increasing only slightly. By 1994, the flash market is expected to exceed $1 billion (Figure 1). The primary reasons for this growth are well acknowledged by customers: flexibility, nonvolatility, low power and cost-effectiveness. As a result of this seemingly insatiable demand for flash, the memory density treadmill that allowed DRAMs and EPROMs to double every 18 months, has accelerated faster than ever before. Intel's introduction of an 8-Mbit component two years after its 2-Mbit product is evidence of the flash memory's rapid advancement.

Intel's first generation of flash devices, ranging in density from 256-Kbit through 2-Mbit densities, were rapidly adopted into code storage applications which formerly used EPROM or EEPROM. In 1991, Intel introduced the first 1-Mbit Boot Block flash memory in response to direct customer requests. As a result of continuing to improve and develop an innovative and broad product line, Intel held an 85% share of the $130 million 1991 market.

INTEL SERVING TWO PRIMARY APPLICATIONS

Today, Intel flash is serving two major application segments: **updatable code storage** and **solid-state mass storage** (Figure 2). Code and data storage comprise the updatable non-volatile memory applications that require high performance, high density and easy update capability. These applications are infrequently updated when compared to solid-state mass storage applications. In this case, erase/write performance is not as critical as integration and performance requirements. This application segment is served effectively with full chip-erase or boot block products.

The second major application segment is **solid-state mass storage** that requires very high density, automated programming and high performance erase/write capability at a very low cost per bit. Erasing and writing portions of the code or data is much more frequent in solid-state mass storage than in updatable firmware applications. In April of 1992, Intel introduced the 8-Mbit FlashFile component whose architecture is optimized for data file storage. Its symmetrically blocked architecture and automated write/erase features gave programming flexibility for a high-performance, solid-state memory system. The compactness of an 8-Mbit device in a TSOP package allows for high-density flash arrays to be included on a system motherboard as well

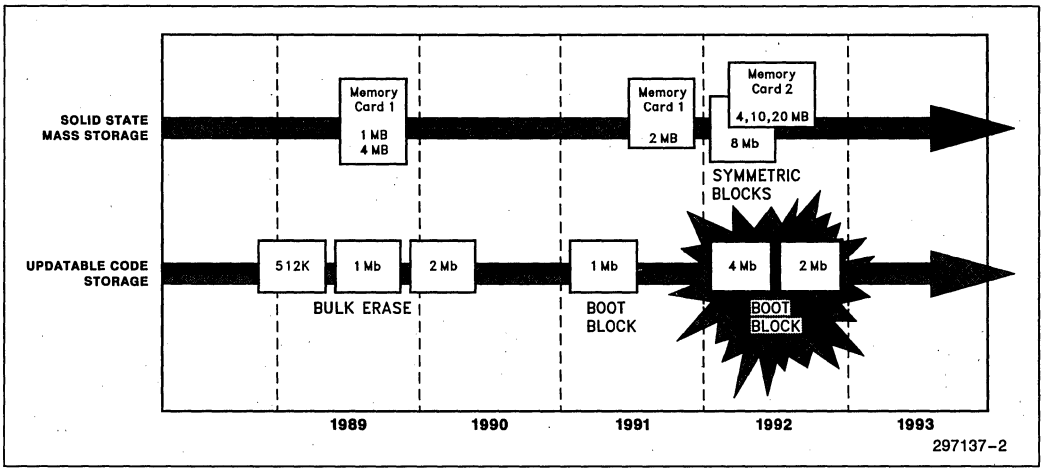


Figure 2

as in memory cards. Intel also offers a second generation of flash memory cards designed to serve the portable mass storage market that are based on the new 8-Mbit FlashFile component. Memory cards add the feature of removability and system upgradability in an industry-standard PCMCIA/ExCA™ format. While flash cards and FlashFile components serve solid-state mass storage applications directly, both can be used in updatable code storage applications as well.

By developing products to fit both of these application segments, Intel strives to serve the needs of a broader base of customer applications with the best nonvolatile memory solutions.

BOOT BLOCK UPDATE

The 28F001BX 1-Mbit Boot Block flash component introduced in June 1991 featured a sectored architecture that has been widely accepted in embedded code storage applications, particularly PC BIOS and cellular communications. Over 20 PC manufacturers use this device in their products, including Compaq, Dell and Zenith Data Systems. The Boot Block architecture gave the manufacturer added flexibility and the ability to differentiate. End users also benefit from the ability to upgrade BIOS software quickly and securely. It is possible for the manufacturer to upload BIOS updates to an electronic bulletin board where the end-user gets the upgrade for the price of the call. The blocked architecture allows the OEM customer to store critical system code securely in the boot block of the device that can minimally bring up the system and download to other locations of the device to initialize the system. The hardware boot locking feature guarantees that even if the power is disrupted during a BIOS update, the system will be able to recover immediately.

The success of the 1-Mbit Boot Block device has resulted in over 150 designs with annual shipments to exceed 1.5 million units worldwide in 1992.

EXTENDING THE BOOT BLOCK PRODUCT LINE: 2- AND 4-MEGABIT FLASH DEVICES

Once the Boot Block architecture became established in the marketplace, customers quickly began to ask for more features and enhancements: speed, density, low power, surface-mount options and an industry-standard upgrade path. These requests were based on the need to expand features in portable computing and communication products.

The 2-Mbit 28F200BX and 4-Mbit 28F400BX are Intel's newest additions to the flash Boot Block product line. These products offer 60 ns performance, two surface-mount packages, and a Boot Block architecture

similar to the 1-Mbit Boot Block device: one lockable Boot Block, two parameter blocks, and the balance of the device is divided into main blocks. The Boot Block securely stores the basic boot code required to initialize the system that can be protected by the hardware-locking feature, ensuring basic system operating code protection. The two parameter blocks can be used for a variety of purposes: manufacturing product code, setup parameters and storing frequently updated data such as system diagnostics. The 28F200BX contains one 16 Kbyte Boot Block, two 8 Kbyte parameter blocks, one 96 Kbyte and one 128 Kbyte main block (Figure 3a). The 28F400BX contains all of the blocks of the 28F200BX plus two additional 128 Kbyte main blocks (Figure 3b). Top and bottom Boot Block versions are available for both densities. Both devices are in the x16/x8 user-selectable organization of the industry-standard, ROM-compatible pin-out in 44-lead PSOP surface mount package. This pinout and package allow an easy upgrade from 2-Mbit to 4-Mbit since only one address is added at the top of the device package. Forty-lead TSOP x8-only versions are also available for both of these devices, providing high density in the smallest form factor. A 56-lead TSOP x16/x8 version will be available for applications requiring x16 organization in a TSOP package.

APPLICATION REQUIREMENTS—PCs

Notebook computer manufacturers today are competing to produce the lightest weight, slimmest and longest battery life products possible. Minimizing board size via reduced chip count is the first improvement the new flash devices provide. BIOS in portable systems has grown beyond the 1-Mbit density to accommodate more sophisticated power management and additional system features, such as PCMCIA card slots. For example, the i386™ SL architecture allows the expansion of BIOS beyond 128 Kbytes (1-Mbit) with internal registers for hardware paging. A new generation of BIOS support for low-power portable computers has been developed by SystemSoft and other vendors which supports the 28F200BX and the 28F400BX. In PC BIOS applications, the main blocks of the Boot Block devices are used for power management code, video drivers, and ROM-executable code, such as MS-DOS*. The arrangement of the blocks for PC BIOS applications based on Intel i386 and i486™ microprocessors is with the Boot Block on top. Other microprocessors and microcontrollers, such as the Intel 80960KX/SX and the Motorola 68000 series, use the bottom Boot Block version of the memory map. Many PC manufacturers have also moved to a x16 organization of their BIOS for

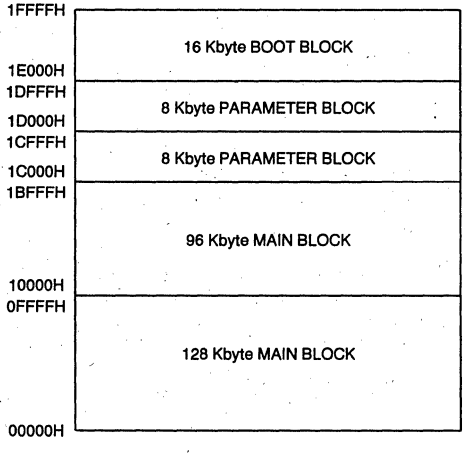


Figure 3a. 28F200BX-Top Boot

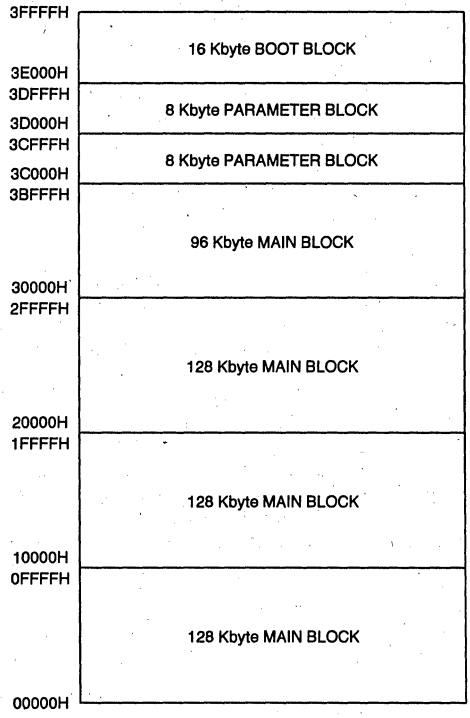


Figure 3b. 28F400BX-Top Boot

higher performance and are using 2-Mbit flash components in parallel to accomplish x16 performance. Other manufacturers have decided to forego x16 and use the 1-Mbit Boot Block and a 1-Mbit bulk-erase 28F010 flash memory to contain their power management code. The 28F200BX solves both of these problems in a compact, single-chip solution. The 28F400BX provides a higher density solution for high performance systems today, as well as a future upgrade for systems currently requiring only 2 megabits.

APPLICATION REQUIREMENTS— TELECOMMUNICATIONS/EMBEDDED

Similar to personal computers, cellular telephone applications use the hardware-lockable block to store basic initialization code to wake up the system and establish basic communication with the host base station. The small parameter blocks are ideal for frequently updated code such as the user's phone directory, re-dial and the activation code necessary to enable user-desired features. The main block contains the code for dynamically managing the cellular communications and executing the voice algorithms. The density requirement for the main code storage has increased significantly in recent years with the conversion to digital cellular transmission, particularly in Europe and Japan. Most cellular designs execute code directly from ROM rather than downloading to RAM. Typical microcontrollers used are: Intel 80C186, Zilog Z8080 and Siemens 80166, all operating in the 8 MHz–13 MHz range. The Motorola 68000 series 16-bit microcontrollers used in some digital cellular designs utilize the x16 and high speed capabilities of these new Boot Block devices. Flash access times ranging from 75 ns to 120 ns produce 0 to 1 wait-state performance. The high-performance 60 ns access time of the 2-Mbit and 4-Mbit Boot Block flash allows zero wait state performance in cellular telephones and other embedded control applications.

Low power has always been a strength of flash due to its inherent nonvolatility, a characteristic that eliminates battery drain required to maintain information stored in volatile RAM when the power is off. When flash is in use, its active power requirement is very low: typical I_{CC} = 35 mA, and standby current is 0.1 mA. The hallmark of this newest generation of flash is low power and high performance. Formerly these features were mutually exclusive, but through advanced circuit development, an automated power saving feature in the device reduces I_{CC} to a low DC level within one access time; 5 mA typical. This significant reduction of the active current time relates directly to the extension of battery life in systems which continuously execute code directly from flash, such as cellular telephones and portable instrumentation.

For optimum system power conservation, future systems are being designed today to operate at 3.3V rather than 5V. Intel will also offer 3.3V versions of the 2-Mbit and 4-Mbit Boot Block devices with I_{CC} active = 10 mA, and t_{ACC} performance = 150 ns.

An alternative design approach of a "5V-only" (programming voltage = operating voltage = 5V) flash memory has been proposed to eliminate the need for a 12V programming supply. While this proposal would eliminate the need for a 12V supply component in a 5V-only environment, it would not provide a low power solution as every flash component would carry the die size cost and power overhead of on-chip voltage pumping. Intel's approach is to first bring the operating voltage to 3.3V, in line with what system designers requested as their first priority for the next generation of portable computers and cellular phones. The focus of lowering the read voltage is also congruent with the usage model for embedded applications of "read often, write few" operation. The 12V programming voltage requirement is more efficiently met with a voltage pump from a vendor such as Maxim. (Inexpensive voltage pumps for 3.3V to 12V are currently available.) The use of a separate voltage pump is significantly more cost-efficient in systems where more than one flash device is used, as cost and space is amortized over multiple devices.

CONCLUSION

Intel continues to serve both updatable non-volatile memory applications as well as the rapidly emerging solid-state mass storage market with solutions tailored to meet their particular needs. The new 2-Mbit and 4-Mbit Boot Block devices offer high performance and low power options for updatable code applications. The Boot Block architecture is compatible with all major microprocessors and microcontrollers. As with all Intel flash products, their low power consumption and small surface mount packaging make the 2-Mbit and 4-Mbit Boot Block flash memories ideal for a wide variety of handheld portable applications.

NOTE:

FlashFile, ExCA, i386 and i486 SL are trademarks of Intel Corporation.

*MS-DOS is a registered trademark of Microsoft Corporation.